Marco, Abed

Node.js

WEEK 1

- What is Node.js?
- Using Node Package Manager (NPM) - PART 1

WEEK 2

- Using require (and import) to include modules

WEEK 3

- Using http to handle http requests and responses
- Using fs to read from and write to files

WEEK 4

- Using Node Package Manager (NPM) - PART 2
- Using Express to make a RESTful API

WEEK 5

- Using process to read arguments from the CLI

Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Why Node.js?

- JavaScript
- Lightweight
- Efficient
- Perfect for data-intensive real-time
- Large ecosystem
- Isomorphic application

Question:

Is node good only for web?

Node.js VS other languages

Python

```
file = open(“testfile.txt”,”w”)

file.write(“Hello World”)
file.write(“This is our new text file”)
file.write(“and this is another line.”)
file.write(“Why? Because we can.”)

file.close()
```

Node.js

```
fs.writeFile('message.txt', 'Hello Node.js', (err) => {
  if (err) throw err;
  console.log('The file has been saved!');
})
```

Getting started with Node.js

---

https://docs.npmjs.com/getting-started/installing-node

```
$ node -v
$ npm -v
```

Question:

In which language is Node written in?

Node Package Manager

---

npm is the package manager for JavaScript and the world’s largest software registry

Modules

Node.js has a simple module loading system. In Node.js, files and modules are in one-to-one correspondence (each file is treated as a separate module).

Modules keep your code reusable.

Week2

- Classes and OOP basics
- Promises as alternative to Callbacks
- Node Package Management
- Babel boilerplate for start using ES6

Object-oriented programming (OOP)

Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.

Class

In object-oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

Class in ES6

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }

  get area() {
    return this.calcArea();
  }

  calcArea() {
    return this.height * this.width;
  }
}
```

```
const square = new Rectangle(10, 10);

console.log(square.area);
```

Contact list as a Class

```
const my_contact_list = new ContactList();

my_contact_list.add({
    name: “Marco”,
    phone: +4552838188
});
```

Question:

What constructor does in a class?

Question:

Is constructor special method necessary?

NPM

Important commands

- npm init
- npm install
- npm uninstall

Important flags

- --save
- --save-dev
- -g

https://docs.npmjs.com/

Node with ES6

Babel is a JavaScript compiler, that allows you to use next gen JavaScript today.

It basically “transform” modern JS to a version that NodeJS can understand.

https://babeljs.io/

Let’s build a boilerplate for use ES6 with NodeJS

Today we are going to learn how to set up babel for node project, but you do not need to do every time, build your own template or use one of the many out there, like mine:

https://github.com/pmcalabrese/node-babel

Node core modules

- Http
- Fs
- Process
- Crypto

https://nodejs.org/dist/latest-v8.x/docs/api/

Callbacks to Promises

Why?

- Avoid callback hell
- Easier flow control

https://developer.ibm.com/node/2016/08/24/promises-in-node-js-an-alternative-to-callbacks/

https://hackernoon.com/node8s-util-promisify-is-so-freakin-awesome-1d90c184bf44

```
function readFileAsync (file, encoding) {
  return new Promise(function (resolve, reject) {
    fs.readFile(file, encoding, function (err, data) {
      if (err) return reject(err) // rejects the promise with `err` as the reason
      resolve(data)               // fulfills the promise with `data` as the value
    });
  });
}

readFileAsync('myfile.txt').then(console.log, console.error);
```

Week3

- Networks
- HTTP
- HTTP Server in Nodejs
- Let’s code an HTTP Server

Networks

A system of computers that are joined together so that they can communicate by exchanging information and sharing resources.

Question:

_ _ _

Name 3 networks that you use every day?

HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

HTTP in Node.js

Node.js is very good framework for networking, it has a very strong HTTP (core) module. And many simple libraries build around it.

https://nodejs.org/api/http.html

HTTP in Node.js

```
const http = require('http');
const fs = require('fs');
const index = fs.readFileSync('index.html');

http.createServer(function (req, res) {
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.end(index);
}).listen(9615);
```

http://www.restapitutorial.com/httpstatuscodes.html

Week 4

- Express vs native http library
- REST API
- Let’s write an HTTP Server with Express
- Gotchas :)

Question:

Why promises are good?

Question:

_ _ _

What HTTP stands for?

Express vs native http library

- HTTP library is **powerful, but it’s not easy to write** a full HTTP Server
- By using an HTTP library for write an HTTP server could end up in a tedious work, and a lot of code. **Lots of code is usually means lots of bugs.**
- By **writing a tiny HTTP Server on your own you learn a lot**.
- **By using libraries you can build powerful software** with few line of code. Taking advantage of hundreds of hours of development done for you.
- **Libraries condense many solutions** for a lot of common problems, very handy

REST API

An architectural style called REST (Representational State Transfer) advocates that web applications should use HTTP as it was originally envisioned. Lookups should use GET requests. PUT, POST, and DELETE requests should be used for mutation, creation, and deletion respectively.

REST API

https://expressjs.com/en/starter/basic-routing.html

HTTP VERBS		Example
GET	Retrieve a resource or a set of resources	GET /car GET /car/122

PATCH	Edit an existing resource	PATCH /car/122
		{
		gears: 5
		}

POST	Create a new resource	POST /car
		{
		brand: “Ford”,
		model: ”Fiesta”
		gears: 5,
		color: “blue”
		}

DELETE	Delete an existing resource	DELETE /car/122

Express

Express

```
$ npm install express --save
```

https://expressjs.com/

The “6 lines Express web server example“

```
const express = require('express')
const app = express()

// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
})


app.listen(3000, function () {
 console.log('Example app listening on port 3000!')
})
```

Week 5

- Express server (REST API + static pages)
- Write a FE that communicate with the Express Server
- * GAME WITH PRICE *
- Deploy our code :)

EXPRESS WEB SERVER

A typical web Express web server

- Serve static pages (html + css + JS + images + more)
- Serve REST API (json)
- Connect to DB

The one difference that you need to remember is that static server serves data but also the representation of your data (HTML CSS IMAGES set), REST API Server contains only data (which can be used from different clients, mobile apps etc). You need to learn both because you need both.

A WEB APP (a full overview on what's coming)

A typical web app has the following parts:

- Frontend (client code)
- Backend (server code)
- Database (persist data)
- Infrastructure (where your backend and db runs)

We need a proper FE

Let’s decide on a **framework**

Game time

Hit us up

marco & r0hitsharma on Slack

Tips:
Share code as snippets on Slack